Desenvolvimento e Automação de Software de Baixa Energia
Refinamento do Tópico de Computação sustentável
Luigi Carro - INF - UFRGS

Resumo
A crescente utilização de recursos computacionais pela sociedade em geral pode significar 21% do consumo mundial de energia em 2030 [1]. Neste cenário, as opções para redução do consumo de energia podem vir por mudanças no hardware ou software. Infelizmente, a diminuição do consumo energético pela melhoria do hardware encontra-se no limite, e a solução para redução do consumo terá de vir pelo desenvolvimento de software de baixo consumo energético. Contudo, as técnicas de software exigidas para redução de energia não são óbvias, e demandam tempo de projeto inaceitável para a velocidade de produção de software requerida pela sociedade atual. Nesta proposta discute-se o desafio de diminuir a energia somente via software, e o desenvolvimento de ferramentas que impeçam um aumento no tempo de produção de software.

Abstract
The increasing use of computing resources by society in general could account for 21% of global energy consumption by 2030 [1]. In this scenario, the options for reducing energy consumption may come from changes in hardware or software. Unfortunately, reducing energy consumption through hardware improvements is limited, and the solution to reducing consumption will have to come from developing low-energy software. However, the software techniques required to reduce energy are not obvious, and demand unacceptable design time for the software production speed required by today's society. This proposal discusses the challenge of reducing energy through software alone, and the development of tools that prevent an increase in software production time.

1)Alinhamento deste desafio com computação sustentável

O consumo de CO2 dos datacenters só aumenta. Pelos últimos 50 anos a solução para diminuição de energia sempre foi aguardar a próxima versão da tecnologia, que com o fim da lei de Moore não pode mais ser usada. Haverá portanto pressão da sociedade para que a computação diminua seu *footprint* de carbono, e também porque o rápido crescimento da demanda energética não poderá ser coberto pela expansão do sistema elétrico brasileiro. Sendo assim, mudanças no software serão a maneira de se conseguir alto desempenho com baixa energia, em diferentes plataformas (mobile e datacenters). As propostas deste documento lidam não somente com um problema brasileiro, mas mundial.

2)Os limites do hardware

Atualmente, software é executado em dois extremos: plataformas mobile, com severas restrições de potência, onde a duração da bateria é um prêmio; e nos datacenters, com CPUs ou GPUs, onde o volume de computadores exige enorme quantidade de recursos para resfriamento, e a energia é extremamente custosa em relação ao preço de funcionamento do

datacenter. Dados recentes apontam que 40% da energia elétrica consumida num datacenter é voltada à refrigeração [2].

Soluções de hardware poderiam ser utilizadas, e o são no telefone celular, por exemplo. Tem-se, pelo limite legal de 3W de potência, um conjunto de aceleradores especializados de vídeo e áudio que são ligados somente quando necessário. Contudo, a arquitetura de um celular é extremamente heterogênea, com diversas CPUs internas que funcionam como aceleradores, e seu desenvolvimento é custoso. Portanto, esperar que o hardware seja a solução é inviável, já que o fim da lei de Moore está muito presente. Para novos nós tecnológicos a potência máxima tem se mantido constante, e os ganhos energéticos são cada vez mais difíceis, como mostra o dark silicon.

A lei de Moore mais fraca significa que o tempo para se dobrar o número de transistores na mesma área deve passar de 2 anos para um período maior. Na figura 1, especulativamente, tem-se curvas de 2, 3 e 4 anos. A realidade estará provavelmente entre as curvas de 2 a 4 anos. Contudo, a desaceleração da lei de Moore é apenas uma variável. Outra variável importante é a velocidade de crescimento de software.

O crescimento do software é uma variável mais difícil de se obter. Por exemplo, há o crescimento em custo, estimado em 12% ao ano. Mas isto não reflete o número de linhas executadas, somente seu impacto. Dados da Nasa sobre o crescimento do número de linhas de código em seus projetos indicam um crescimento de 25% ao ano durante os anos de 1990 até 2010 [3]. Outra fonte facilmente observável é o crescimento do sistema operacional Windows, que passou de 10 milhões de linhas em 1993 para 50 milhões de linhas em 2003 [4]. Nestes 10 anos tem-se também 25% de crescimento composto ao ano.

Como a métrica de número de linhas de código é imperfeita, sobretudo com o uso corrente de linguagens mais abstratas, e portanto com maior capacidade de expressão, para melhor refletir realidade utilizou-se a equação clássica de arquitetura de computadores,
$T=\#I*CPI*Tciclo$ (1).

Em (1) o tempo de execução depende do número de instruções a serem executadas, do número de ciclos médio por instrução, e do tempo de ciclo. Partindo-se de 1 bilhão de instruções, com um ciclo de relógio de 1 GHz (1 segundo de execução numa CPU), pode-se analisar como o impacto do crescimento no número de instruções executadas (CAGR 25%aa) e das melhorias eventuais do hardware (baixa no Tciclo pela lei de Moore) são combinados. Na figura 1 assumiu-se um CPI de 1,5, levando-se em conta os efeitos de cache miss e diferença de velocidade de memórias DRAM, cada vez mais impactantes nos problemas atuais movidos por enorme massa de dados.

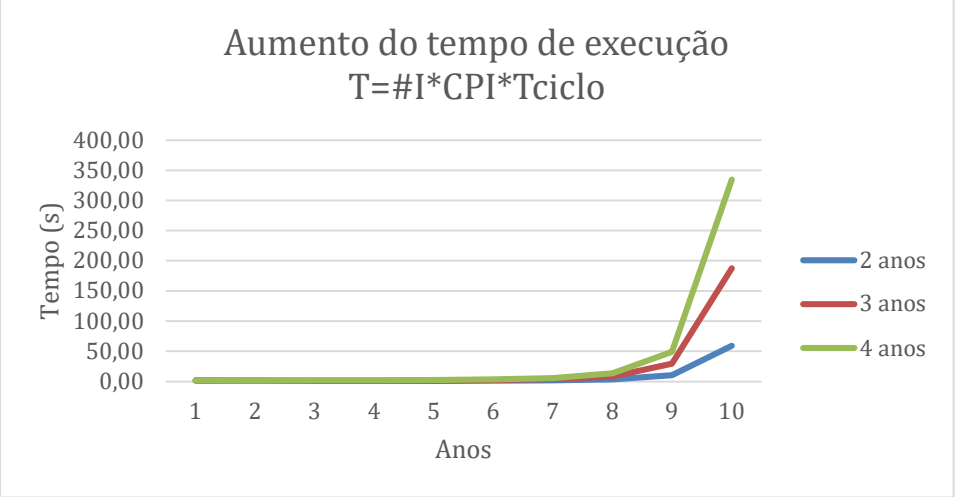

Figura 1 - os efeitos de uma lei de Moore mais fraca junto com o crescimento do número de instruções executada. Fonte: o autor

Na figura 2 tem-se como ambas as variáveis, crescimento do software e velocidade do hardware crescendo mais lentamente, afetarão a energia. Dado que a Energia é expressa por E = P*T (2), e assumiu-se a potência constante de 50W, como limite máximo de dissipação, e usou-se o tempo de computação da figura 1 (3 anos). O efeito é claramente exponencial, já que o crescimento do tempo também o é.

Mesmo arquiteturas novas como GPUs estão submetidas ao mesmo processo. A diferença aqui é que existem *n* processadores a serem usados, com o efeito de se diminuir drasticamente o tempo de execução, desde que a aplicação seja massivamnete paralela, mas com um excesso de potência. Neste contexto tecnológico, a responsabilidade para se obter baixa energia será do software.

3)As necessidades do software e o gap para produção de software de baixa energia

Como já mencionado neste artigo, o crescimento do software implica em milhões de linhas de código. Tem-se hoje o recurso de linguagens mais abstratas de programação, e o uso indiscriminado de APIs, tornando mais produtivo o programador. Na outra ponta, linguagens mais abstratas são obviamente mais distantes do hardware, e portanto é mais difícil para o programador controlar os aspectos que consomem energia (ponto flutuante, cache misses, acesso às memórias DRAM, predição de saltos, etc). Os dados que devem ser processados também sofreram um aumento exponencial na última década, e a tendência é que continuem aumentando. O consumo energético do ChatGPT-3, por exemplo, foi igual ao de uma cidade dinamarquesa de 75 mil pessoas em janeiro de 2023.

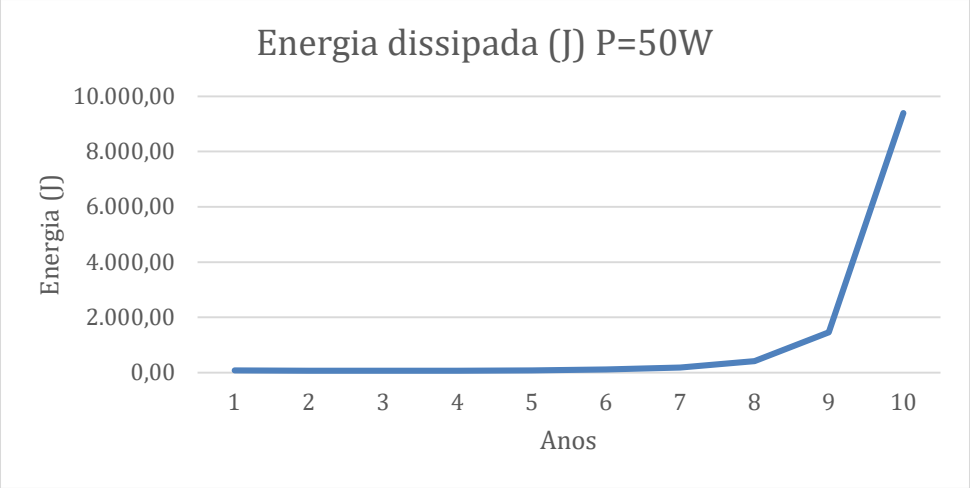

Figura 2 - impacto na energia de uma CPU. Fonte: o autor

A produção de software hoje em dia expõe o programador a uma enorme quantidade de APIs, mas o programador não tem noção do efeito de seu uso no consumo. Além disto, a velocidade exigida para entrega dos produtos de software não diminui, e portanto a pressão por velocidade de entrega não permite uma reflexão, por parte do desenvolvedor, de como atingir mínimos energéticos.

As necessidades de abstração e manutenção do software moderno acarretam mais consumo. Por exemplo, em [5] demonstra-se como uma ferramenta que auxilia na abstração provoca aumento de consumo de energia de até 65%. Embora existam algumas estratégias para diminuição de consumo em programas, largamente estudadas no âmbito de sistemas embarcados, todas elas implicam em alto tempo de projeto. O resultado então é uma impossibilidade: precisa-se de mais ferramentas para se desenvolver software de milhões de linhas de código, mas estas ferramentas aumentam o consumo, e ainda não há estratégias consolidadas de aplicação de técnicas de redução de consumo por parte do programador, ou de ferramentas que ajudem na diminuição do consumo energético sem prejuizo do tempo de projeto.

Some-se a este cenário a falta de treinamento de programadores. Como reduzir em 50% o consumo de um programa? Esta é uma pergunta para a qual os estudantes não foram treinados adequadamente, e há no mercado mundial poucos profissionais capazes de responde-la.

4)O desafio em si

O desafio pode ser dividido em duas partes. Na primeira, tem-se de desenvolver técnicas de redução de energia em software para aplicações-chave. Por exemplo, em [6] obteve-se uma redução de 96 vezes no consumo de CNNs grandes, somente usando técnicas de software. Em [7] foram estudados diferentes implementações de árvores e *forest-trees*, e observou-se que árvores diferentes demandam soluções diferentes, com ganhos de energia de até 4x, puramente pela escolha do algoritmo correto. Tem-se então a segunda parte do desafio: além da produção de algoritmos que possam economizar energia, tem-se de automatizar seu uso, que será função

do tipo de dado com que se está trabalhando. Este, na verdade, é um desafio mundial da Computação, não apenas brasileiro.

5)Possíveis soluções e métricas e avaliação

Algumas soluções já estão em andamento. Por exemplo, os banco de dados vetoriais partem de um modelo diferente, para melhor usar o hardware massivamente paralelo que se encontra à disposição na forma de GPUs. Embora a motivação inicial tenha sido desempenho, pela extrema velocidade de execução da multiplicação de matrizes a energia fica reduzida.

Além dos estudos em [6-7], há 2 anos o proponente oferece uma disciplina opcional de Software de Baixa energia, onde diferentes estratégias são apresentadas aos alunos. Pretende-se ampliar o esforço na produção de artigos, livros e material didático, que possa ser distribuído pelo país, para que mais alunos tenham contato com o tema. O ponto de "pensar sobre o assunto" já foi atingido, mas deve-se escalar a pesquisa na área. A implementação de datacenters no Brasil, e a chamativa economia pelo uso adequado de software podem ser ótimos catalisadores.

Quanto a métricas de avaliação, estas seriam relativamente fáceis de serem obtidas, pelo número de pessoas capazes de pensar o software de baixa energia, e pela implementação do mesmo, medindo-se o desempenho de datacenters no país.

Referências

[1] N. Jones. How to stop data centres from gobbling up the world's electricity. Nature, 561(7722):163–167, 2018.

[2] https://www.digitalrealty.com/resources/articles/future-of-data-center-cooling Último acesso em 08/09/2024

[3] Butler, R.; Pennotti, M. The evolution of software and its impact on complex system design in robotic spacecraft embedded systems. Dec 2013, Procedia Computer Science.

[4] https://en.wikipedia.org/wiki/Source_lines_of_code Último acesso em 08/09/2024

[5] Calero. C.; Polo, M.; Moraga, A. Investigating the impact on execution tiem and energy consumption of developing with Spring. Sustainable Computing: Informatics and Systems. 2021, Elsevier.

[6] Gonçalves, L.R.; Moura, R.F.; Carro, L. Aggressive energy reduction for video inference with software-only strategies. ACM transactions Embedded Computer Systems, v.18, issue 5, p. 1-20, 2019.

[7] Bergozzi, V.; Carro, L. Low energy decision trees. Relatório interno.